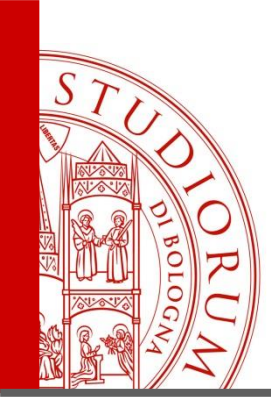


Università degli Studi di Bologna
Corso di Laurea in Ingegneria Informatica

Progettazione

Ingegneria del Software T

Prof. MARCO PATELLA
Dipartimento di Informatica – Scienza e Ingegneria (DISI)

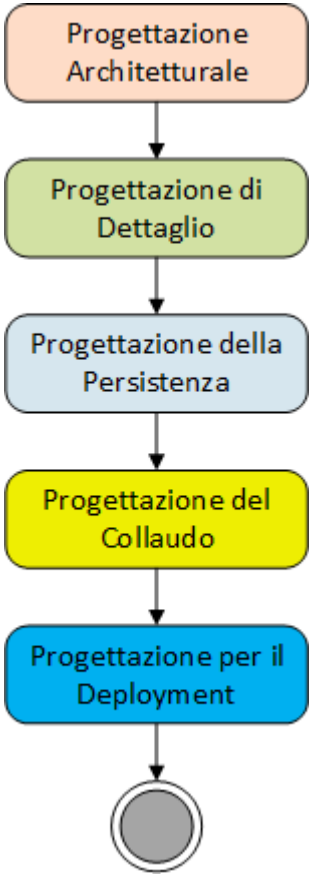
Sommario

- Progettazione
- Progettazione Architetturale
- Progettazione di Dettaglio
- Progettazione della Persistenza
- Progettazione del Collaudo
- Progettazione per il Deployment

Progettazione

- Obiettivo:
  attraverso una serie di raffinamenti successivi dell'Architettura Logica arrivare ad ottenere l'Architettura del Sistema
  Vanno considerati anche tutti gli aspetti vincolanti che sono stati trascurati nelle fasi precedenti
  - Questa fase deve mirare non solo a individuare e descrivere una soluzione al problema (*what/how*), ma soprattutto a descrivere i motivi (*why*) che l'hanno determinata
- Risultato:
  - Architettura del Sistema
  - Schema Persistenza
  - Piano finale del Collaudo
  - Indicazioni per il Deployment

Progettazione



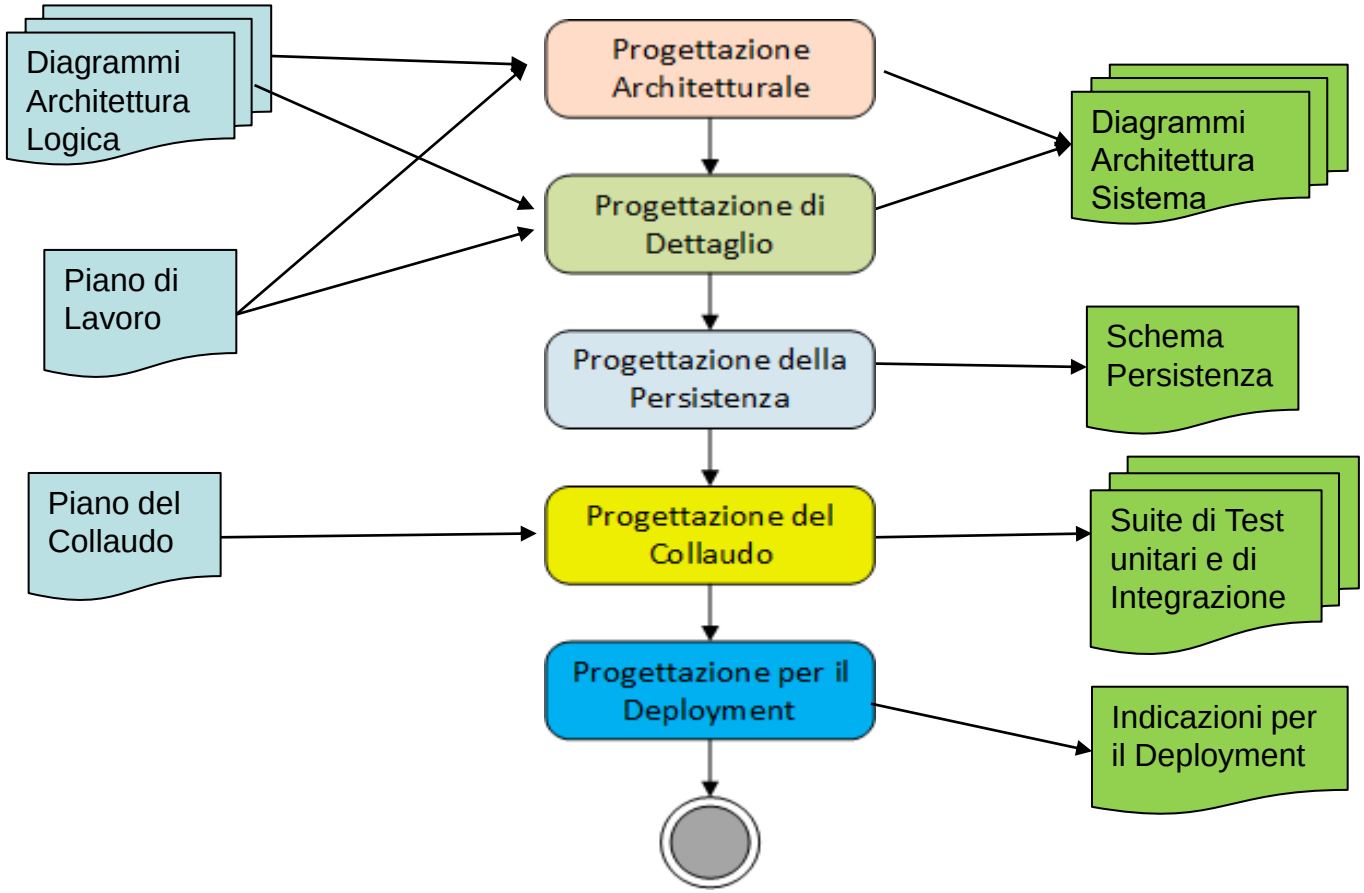
Progettazione

1. Progettazione Architetturale
   Obiettivo: definire l'Architettura del Sistema tenendo conto di tutti i vincoli e delle forze in gioco
2. Progettazione di Dettaglio
   Obiettivo: progettare nel dettaglio ogni aspetto del Sistema
3. Progettazione della Persistenza
   Obiettivo: progettare i meccanismi per la persistenza dei dati

Progettazione

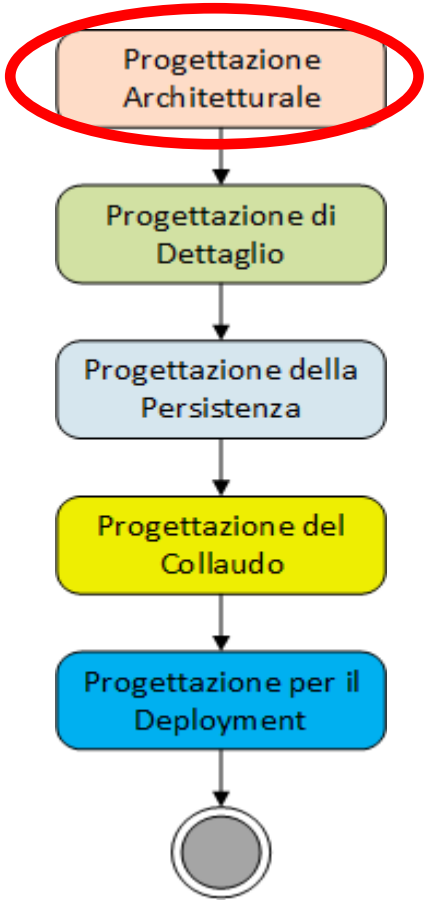
4. Progettazione del Collaudo
   Obiettivo: definire in modo chiaro e preciso come il sistema dovrà essere collaudato una volta terminata l'implementazione
5. Progettazione per il Deployment
   Obiettivo: progettare il sistema in modo da rendere semplice il deployment sulle macchine e per garantire la sicurezza

Progettazione



Progettazione architetturale

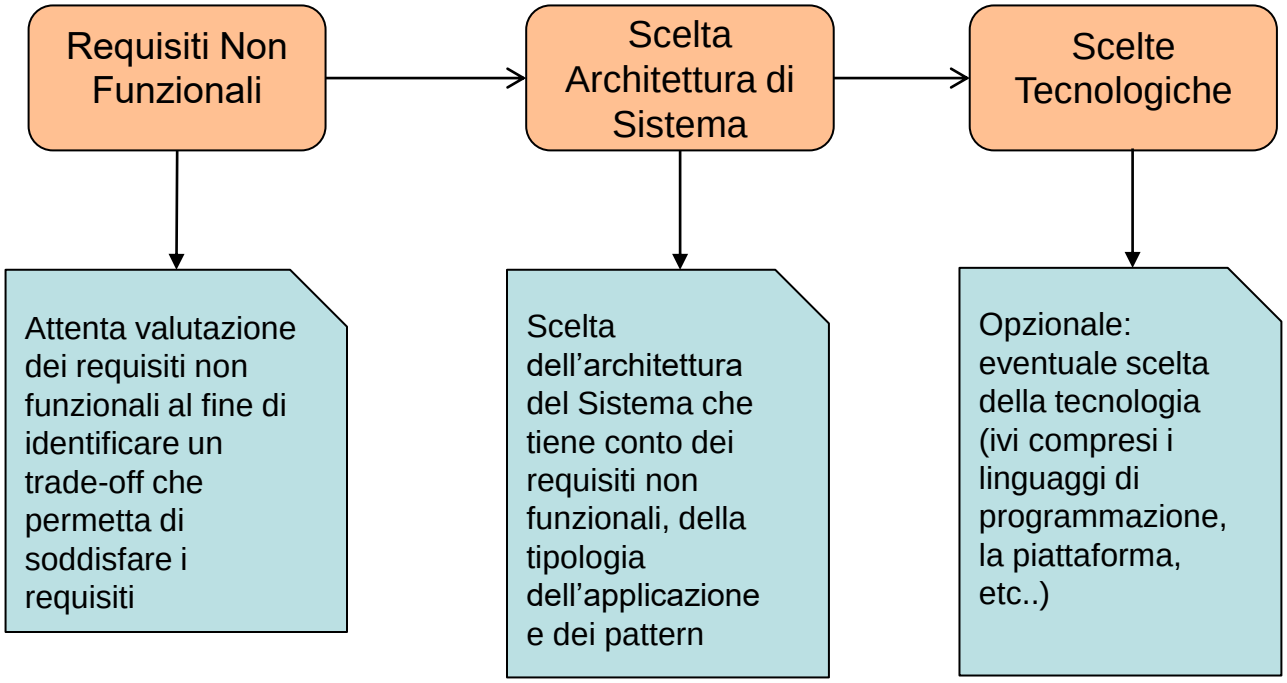
Progettazione



Progettazione Architetturale

- Nella Progettazione Architetturale gli ingegneri devono prendere delle decisioni che influenzano profondamente il sistema
- Basandosi sulle proprie esperienze e conoscenze devono rispondere ad alcune domande fondamentali:
  1. C'è un'architettura applicativa generica che può essere utilizzata come modello per il sistema che sto progettando?
  2. Come sarà distribuito il sistema tra più processori?
  3. Quale stile o quali stili sono adatti al sistema?
  4. Quale sarà l'approccio fondamentale utilizzato per strutturare il sistema?
  5. Come saranno scomposte in moduli le unità strutturali del sistema?
  6. Quale strategia sarà usata per controllare l'operato delle unità del sistema?

Progettazione Architetturale



Requisiti Non Funzionali

- L'architettura del sistema influenza
  - *le prestazioni,*
  - *la robustezza,*
  - *la distribuibilità*
  - *la manutenibilità*

  di un sistema
- La struttura dell'architettura tipicamente è condizionata
  - dalla tipologia di applicazione che si vuole realizzare
  - dai requisiti non funzionali

Requisiti Non Funzionali

- Se le ***prestazioni*** sono un requisito critico l'architettura dovrebbe essere progettata
  - localizzando le operazioni critiche all'interno di un piccolo numero di componenti
  - minimizzando le comunicazioni possibile tra essi
- Questo porta a dover definire componenti "grandi" per ridurre la comunicazione

Requisiti Non Funzionali

- Se la ***protezione dei dati (security)*** è un requisito critico l'architettura dovrebbe essere progettata
  - con una struttura "stratificata"
  - collocando le risorse più critiche nello strato più interno e protetto
- Questo porta a dover definire una struttura con un alto livello di convalida di protezione a ogni strato
- NB: Quando si valuta l'aspetto della protezione dei dati tenere conto di tutte le indicazioni che sono emerse nella parte della Security Engineering

Requisiti Non Funzionali

- Se la ***sicurezza (safety)*** è un requisito critico l'architettura dovrebbe essere progettata
  - in modo tale che le operazioni relative siano tutte collocate in un singolo componente o in un piccolo insieme di componenti
  - riduzione dei costi e dei problemi di convalida della sicurezza, possibilità di poter fornire sistemi di protezione correlati
- Questo porta a dover definire componenti "grandi" per localizzare le operazioni

Requisiti Non Funzionali

- Se la ***disponibilità*** è un requisito critico l'architettura dovrebbe essere progettata
  - per comprendere componenti ridondanti
  - in modo che sia possibile sostituirli e aggiornarli senza fermare il sistema
- Questo porta a dover sviluppare un numero maggiore di componenti rispetto a quelli strettamente necessari

Requisiti Non Funzionali

- Se la ***manutenibilità*** è un requisito critico l'architettura dovrebbe essere progettata
  - usando componenti piccoli, atomici, autonomi
  - che possano essere modificati velocemente
  - i produttori di informazione dovrebbero essere separati dai consumatori e le strutture dati condivise dovrebbero essere evitate
- Questo porta a dover sviluppare componenti di piccole dimensioni

Requisiti Non Funzionali

- Ci sono dei conflitti potenziali tra alcune di queste architetture così come abbiamo visto sussistono conflitti tra i requisiti non funzionali
- Esempio: usare componenti “grossi” migliora le prestazioni ma peggiora la manutenibilità e viceversa
- Se sono entrambi requisiti critici occorre trovare un compromesso

Esempio

REQUISITI NON FUNZIONALI

Nell'Analisi del Problema (Tabella Vincoli) sono emersi tre requisiti non funzionali che impongono dei vincoli al sistema:

- Tempo di risposta
- Usabilità
- Sicurezza

Nello specifico caso in esame, Usabilità e Sicurezza hanno pochi conflitti a parte l'eventuale richiesta di un ulteriore login se per caso scade la sessione di lavoro. L'Usabilità impatta molto di più la struttura delle Interfacce che andranno progettate in modo tale da mantenere nelle stesse View le informazioni necessarie alle funzionalità richieste.
Diversa la questione che riguarda Tempo di risposta e Sicurezza, aggiungere strati (layer) e meccanismi di cifratura per migliorare la sicurezza ovviamente porta ad un peggioramento delle prestazioni del sistema, occorre quindi trovare un bilanciamento tra i due aspetti.
Considerando la tipologia di sistema che deve essere sviluppato, si ritiene maggiormente critico l'aspetto di sicurezza dei dati in quanto la "Tabella Valutazione Beni" mette in luce che nel caso di attacchi al sistema andati a buon fine si rischia un'esposizione molto alta con perdite finanziarie e di immagine. Inoltre, gli utenti principali di tale sistema sono operatori umani che spesso non sono in grado di percepire se il Sistema impiega qualche frazione di secondo in più o in meno nella risposta, non si hanno vincoli real-time da soddisfare.

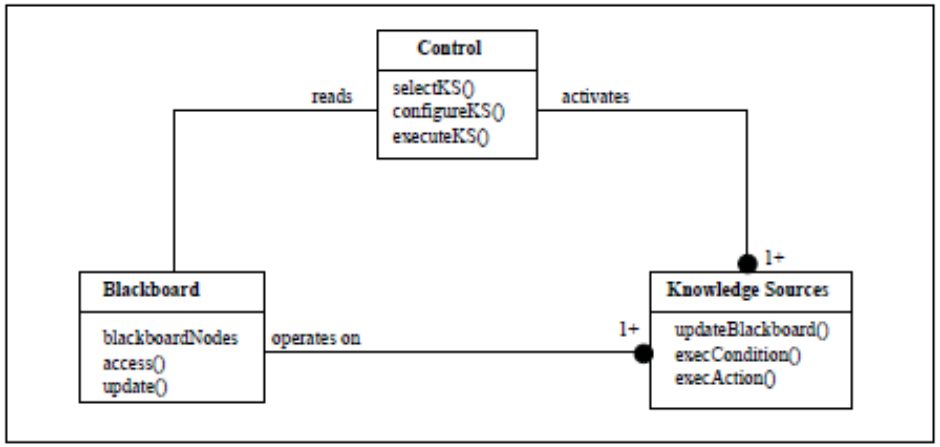
Scelta Architettura

- La scelta dell'Architettura del Sistema deve basarsi su:
  - Architettura Logica definita in fase di Analisi del Problema
  - Trade-off requisiti non funzionali
  - Tipologia di applicazione che si intende sviluppare
  - Adozione di Pattern Architetturali
    - Blackboard
    - MVC/BCE
    - Layers
    - Client/Server
    - Broker
    - Pipe & Filters
    - …

Pattern architetturali "Pattern-Oriented Software Architecture"

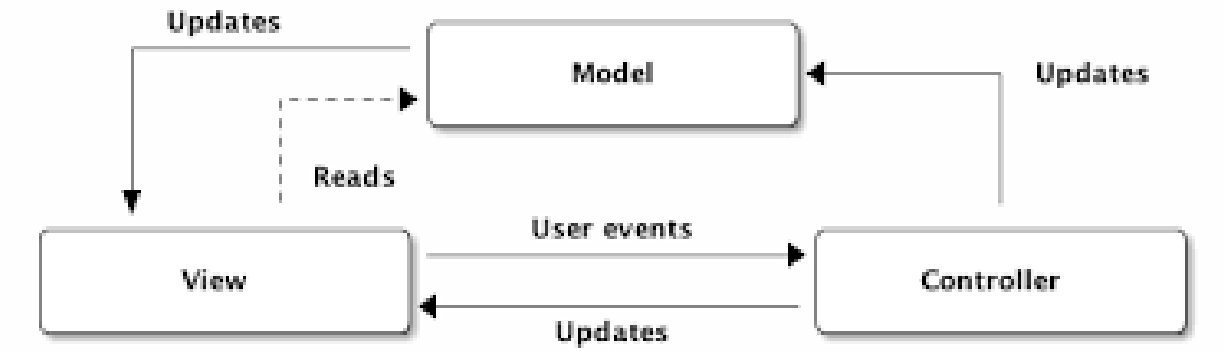
Blackboard

- Il pattern Blackboard aiuta a strutturare quelle applicazioni in cui vengono applicate strategie di soluzione non deterministiche (tipici problemi di intelligenza artificiale)
- I diversi sotto-sistemi condividono la stesse conoscenze attraverso la Blackboard al fine di costruire una soluzione approssimata o parziale

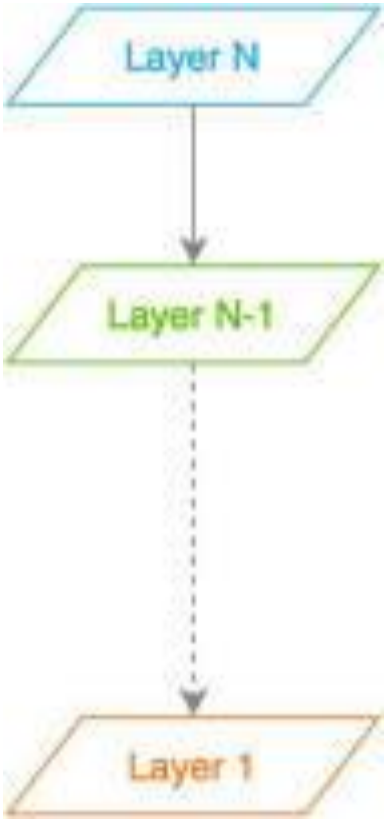


MVC

- Il pattern MVC divide le applicazioni in tre distinte parti:
  - Il model che gestisce i dati
  - Il controller che manipola i dati
  - La view che mostra i dati



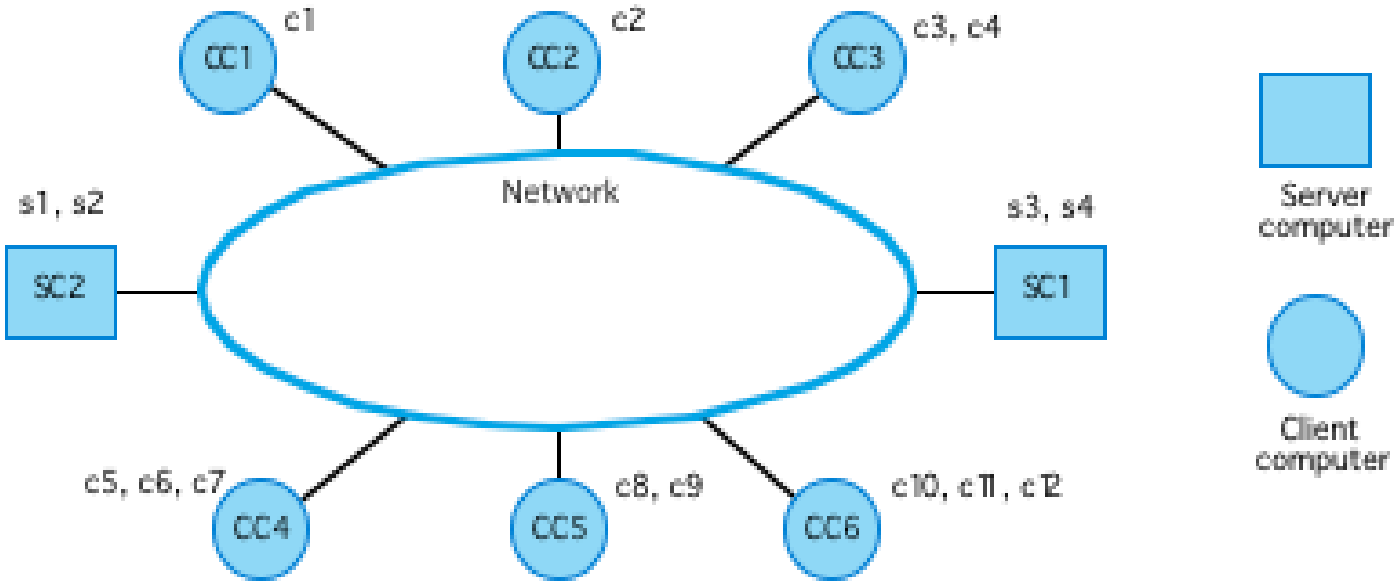
Layer



- Il pattern Layer aiuta a strutturare quelle applicazioni che possono essere scomposte in gruppi di sotto-attività in cui ciascun gruppo si trova a un ben definito livello di astrazione

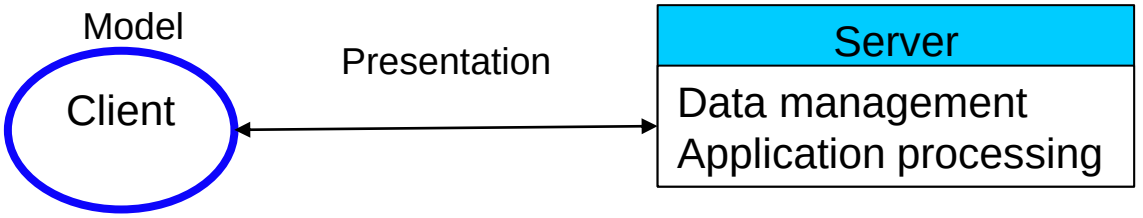
Client/Server

- Il pattern client/server aiuta a strutturare un'applicazione come un insieme di servizi forniti da uno o più server e un insieme di client che utilizza tali servizi



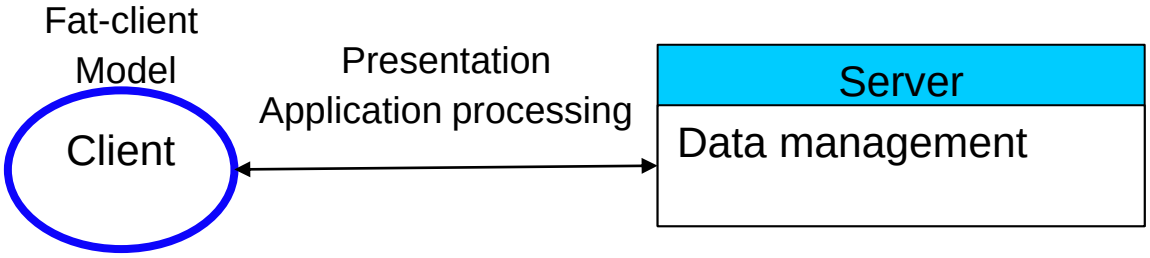
Client/Server

Thin-client Model

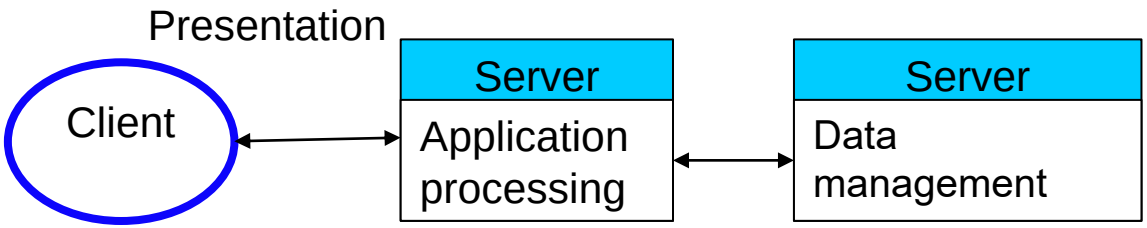


Tutto a carico del server

Fat-client Model



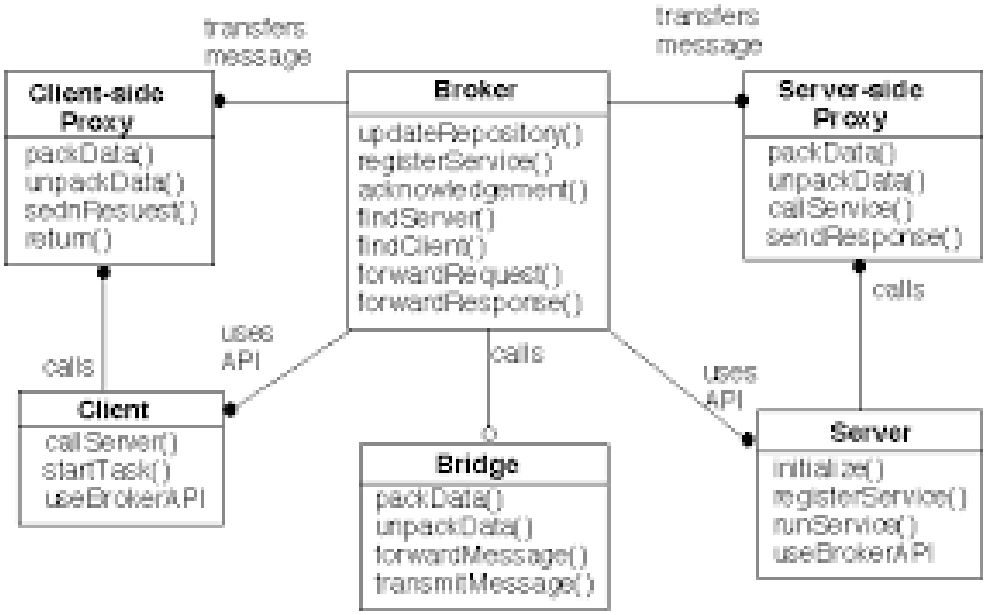
Il server si occupa solo dei dati, il resto lo fa il client



Client/server a 3 Livelli

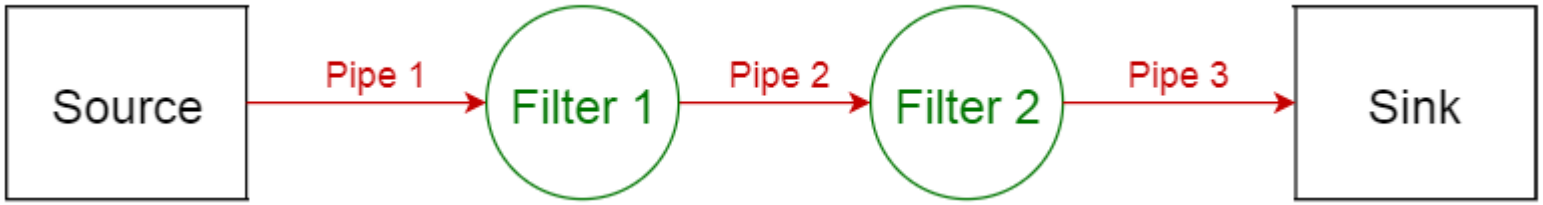
Broker

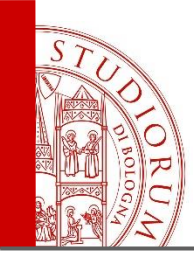
- Il pattern Broker può essere usato per strutturare sistemi distribuiti con un disaccoppiamento tra i diversi sotto-sistemi che comunicano tra loro attraverso remote server invocation
- Il Broker è responsabile della coordinazione delle comunicazioni, come inoltro richieste, invio risposte ed eccezioni



Pipe & Filters

- Il pattern Pipe & Filters aiuta a strutturare quelle applicazioni che processano flussidi dati
- Ogni passo del processo è incapsulato in un apposito filtro e i dati attraversano una pipe di filtri
- Variando l'ordine dei filtri si possono ottenere diversi tipi di sistemi

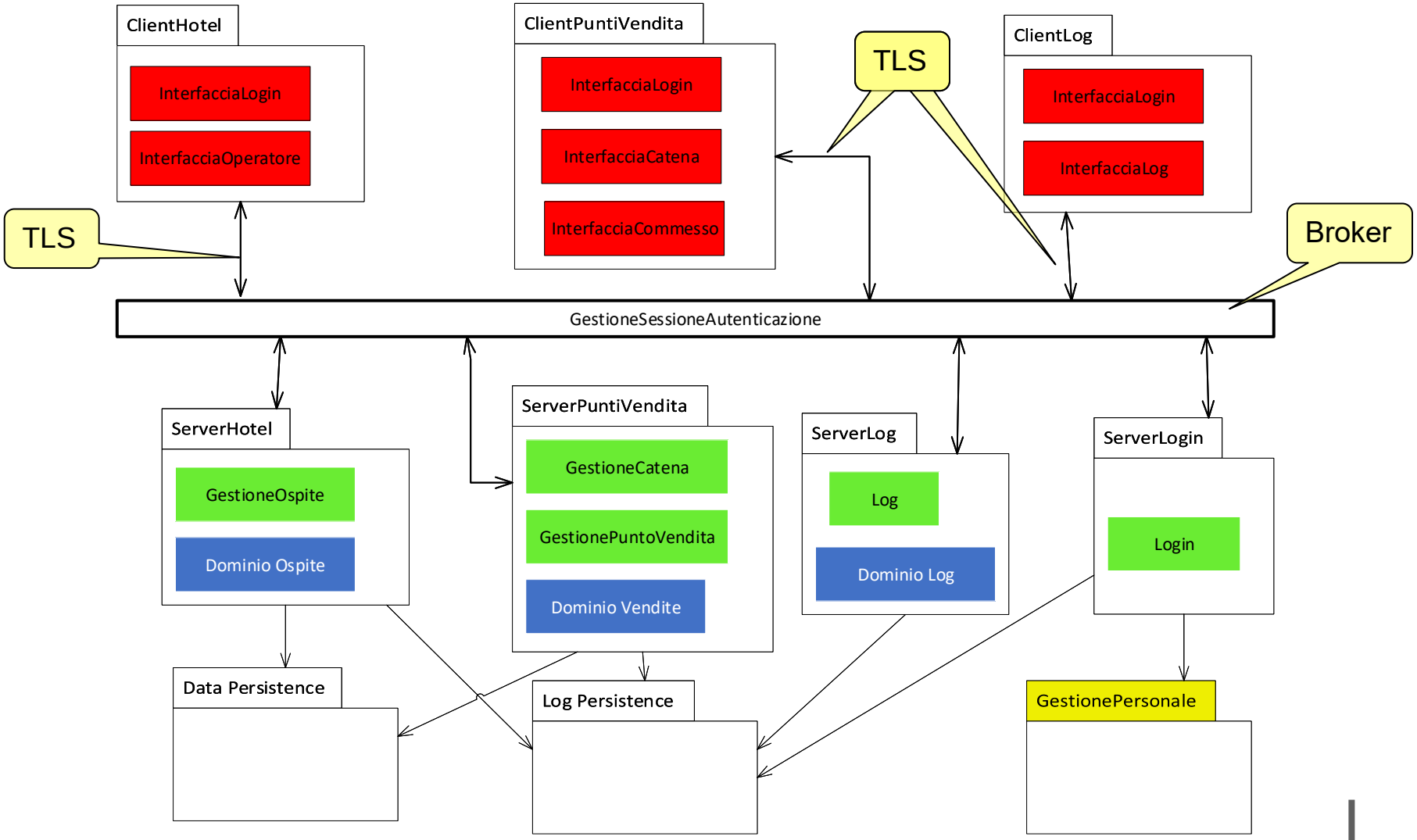




Scelta Architettura

- Come non esiste un processo di sviluppo ideale, non esiste un'Architettura ideale sempre utilizzabile
- Talvolta è necessario usare stili architetturali diversi per parti diverse del sistema al fine di soddisfare tutti i vincoli imposti dai requisiti
- L'adozione dei pattern architetturali può aiutare a trovare il giusto compromesso tra tutte le forze in gioco

Esempio: Villaggio Turistico



Scelte Tecnologiche

- L'uso di una specifica tecnologia (intesa anche come linguaggio di programmazione, piattaforma, strumento, etc.) non è sempre neutro
- In taluni casi potrebbe risultare vantaggioso scegliere le tecnologie già in fase di progettazione **legando così il progetto** alla specifica tecnologia
- Nel caso si decida di scegliere la tecnologia in fase di progettazione, va specificato chiaramente e va fatta un'analisi costi/benefici
- Vanno attentamente studiate le parti della tecnologia adottata in modo che sia poi possibile inserirle nei diagrammi di progettazione

Progettazione di dettaglio

Progettazione

Progettazione di Dettaglio

- La Progettazione di Dettaglio definisce il dettaglio dell'Architettura del Sistema nelle sue tre viste:
  - Struttura
  - Interazione
  - Comportamento
- Per realizzare un sistema funzionante, occorre considerare GUI, DB, *Framework*, librerie, componenti, modifiche al modello per avere software estensibile e modulare…
- È compito della Progettazione di Dettaglio identificare e definire altre classi in accordo alla specifica architettura scelta

Progettazione di Dettaglio

- Durante la Progettazione di Dettaglio, i modelli prodotti nell'Analisi devono essere estesi al fine di progettare i quattro layer principali che compongono il sistema
  - APPLICATION LOGIC – logica dell'applicazione e controllo degli altri componenti
  - PRESENTATION LOGIC – gestione dell'interazione con l'utente a livello logico
    nuovi oggetti: finestre, menù, bottoni, *toolbar*, …
  - DATA LOGIC – gestione dei dati che il sistema deve manipolare
  - MIDDLEWARE – gestione dell'interazione con i sistemi esterni, con la rete e tra i sotto-sistemi

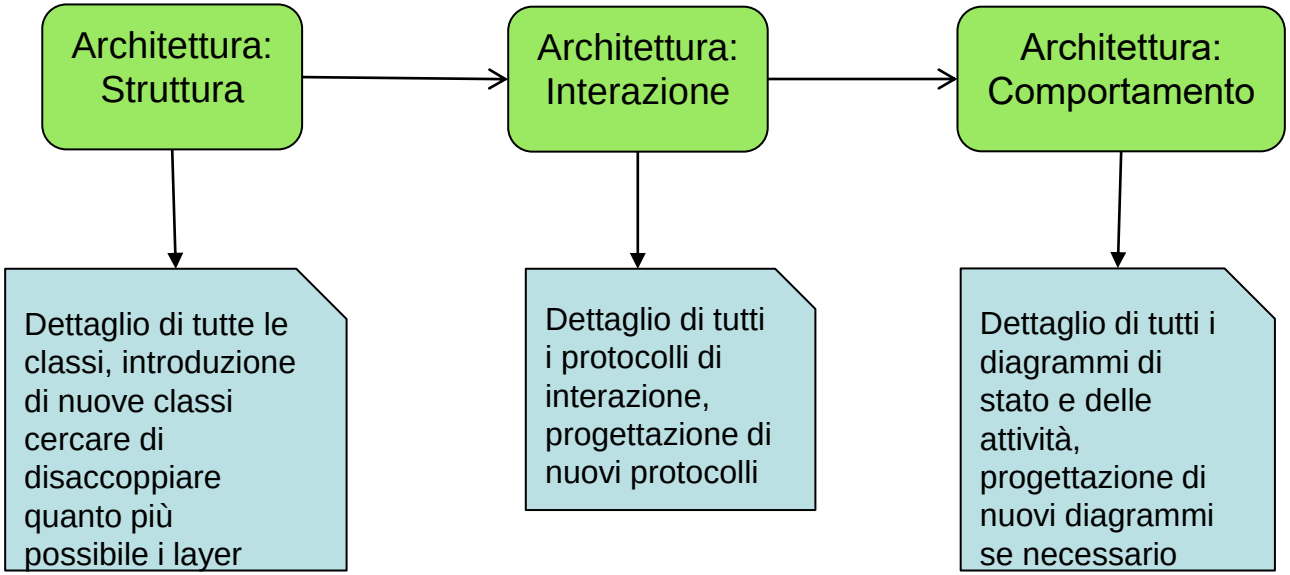
Progettazione di Dettaglio

- Durante la Progettazione di Dettaglio,
  i modelli di Analisi devono essere modificati al fine di:
  - definire in dettaglio le classi e delle loro relazioni
  - supportare caratteristiche specifiche per comunicazioni, diagnostica, protezione dei dati,…
  - riuso di classi e/o componenti disponibili
  - miglioramento delle prestazioni
  - supporto alla portabilità
  - …

Progettazione di Dettaglio

- Massima indipendenza possibile da
  - Linguaggio (e ambiente) di programmazione
  - DBMS
  - Sistema Operativo
  - Hardware
- Le caratteristiche specifiche del contesto utilizzato devono essere tenute in conto solo se
  - sono vincolanti (requisiti non funzionali)
  - si è esplicitamente scelto di legarsi a una tecnologia nella progettazione architetturale

Progettazione di Dettaglio



Architettura: Struttura

- Durante la Progettazione di Dettaglio della parte di Struttura è necessario definire
  - tipi di dato che non sono stati definiti in precedenza
  - navigabilità delle associazioni tra classi
  - strutture dati necessarie per l'effettiva implementazione del sistema
  - operazioni che non erano emerse durante la fase di Analisi del Problema
  - eventuali nuove classi necessarie per il corretto funzionamento del sistema

Architettura: Struttura

- Attenzione alla presenza dei “Sistemi Esterni” individuati in fase di Analisi del Problema
- Se nella tabella “Tabella dei Sistemi Esterni” era stato individuato un problema nel “Livello di Protezione” e il Sistema Esterno non risulta avere il livello di sicurezza minimo richiesto occorre applicate il ***pattern Adapter***
  - si ***ingloba (wrappa)*** il Sistema Esterno in una nostra struttura
  - si progetta la struttura in modo tale che soddisfi i livelli minimi di sicurezza richiesti

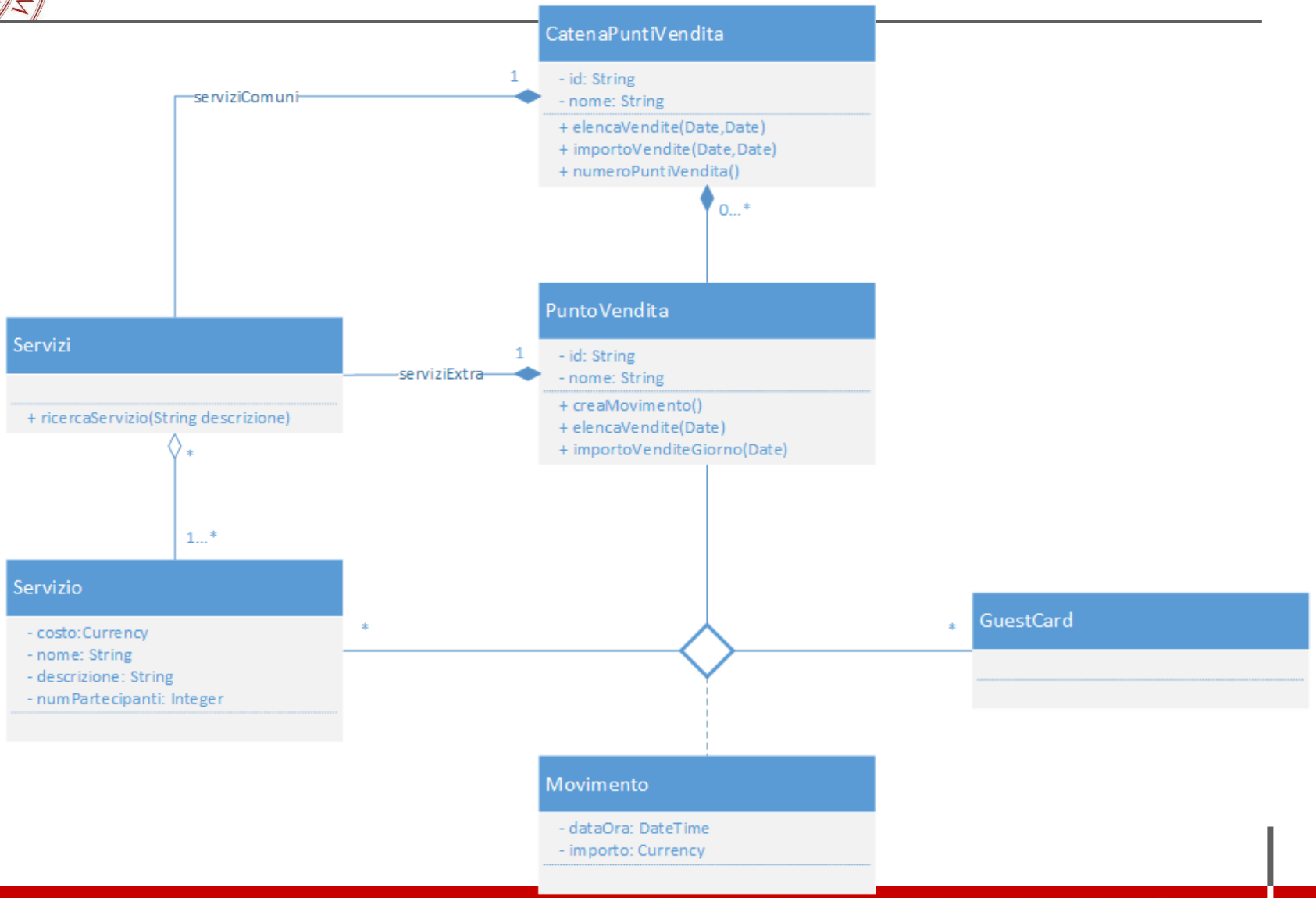
Architettura: Struttura

- Attenzione se si è deciso di vincolarsi a una specifica tecnologia
- Va condotta una attenta analisi e valutazione del livello di protezione offerto dalla tecnologia scelta
- Se tale livello non risulta essere quello minimo richiesto dall'applicazione occorre progettare specifiche parti del sistema per prevenire i buchi di sicurezza legati alla specifica tecnologia
- Ove possibile cercare di applicare il pattern Adapter

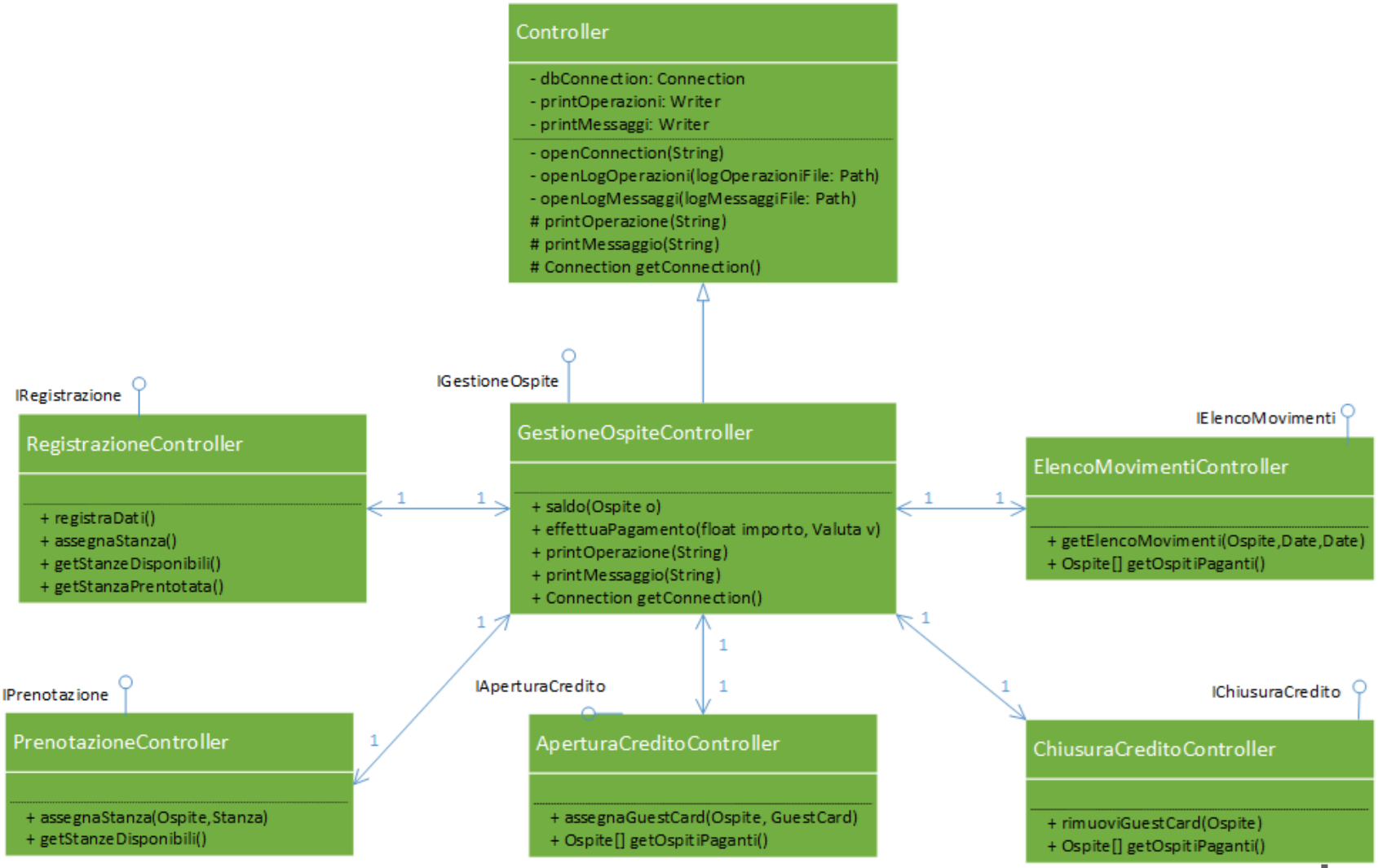
Architettura: Struttura

- Applicazione dei design pattern al fine di realizzare software di qualità facilmente estensibile e modulare
- Applicazione dei principi di progettazione con particolare attenzione al "Dependency Inversion Principle"
- Disaccoppiare i layer del sistema porta molti vantaggi
  - possibile cambiare implementazione di parti del sistema senza che la modifica si ripercuota sulla restante parte → ***design for change***
  - possibile cambiare l'aspetto grafico anche variando la tecnologia realizzativa senza dover modificare l'application logic
  - facile inserire nuove funzionalità con impatto minimo sul sistema

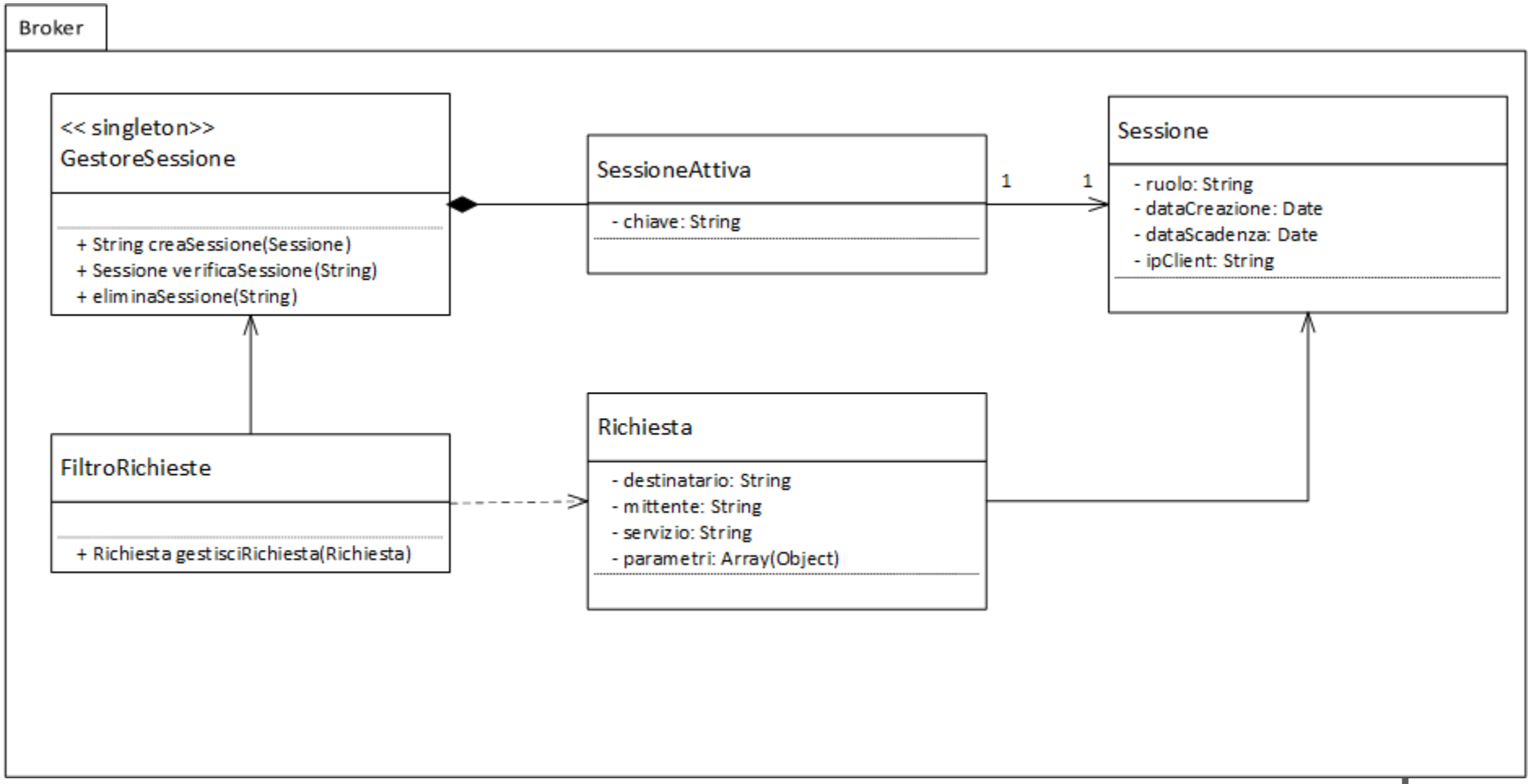
Struttura: Esempio



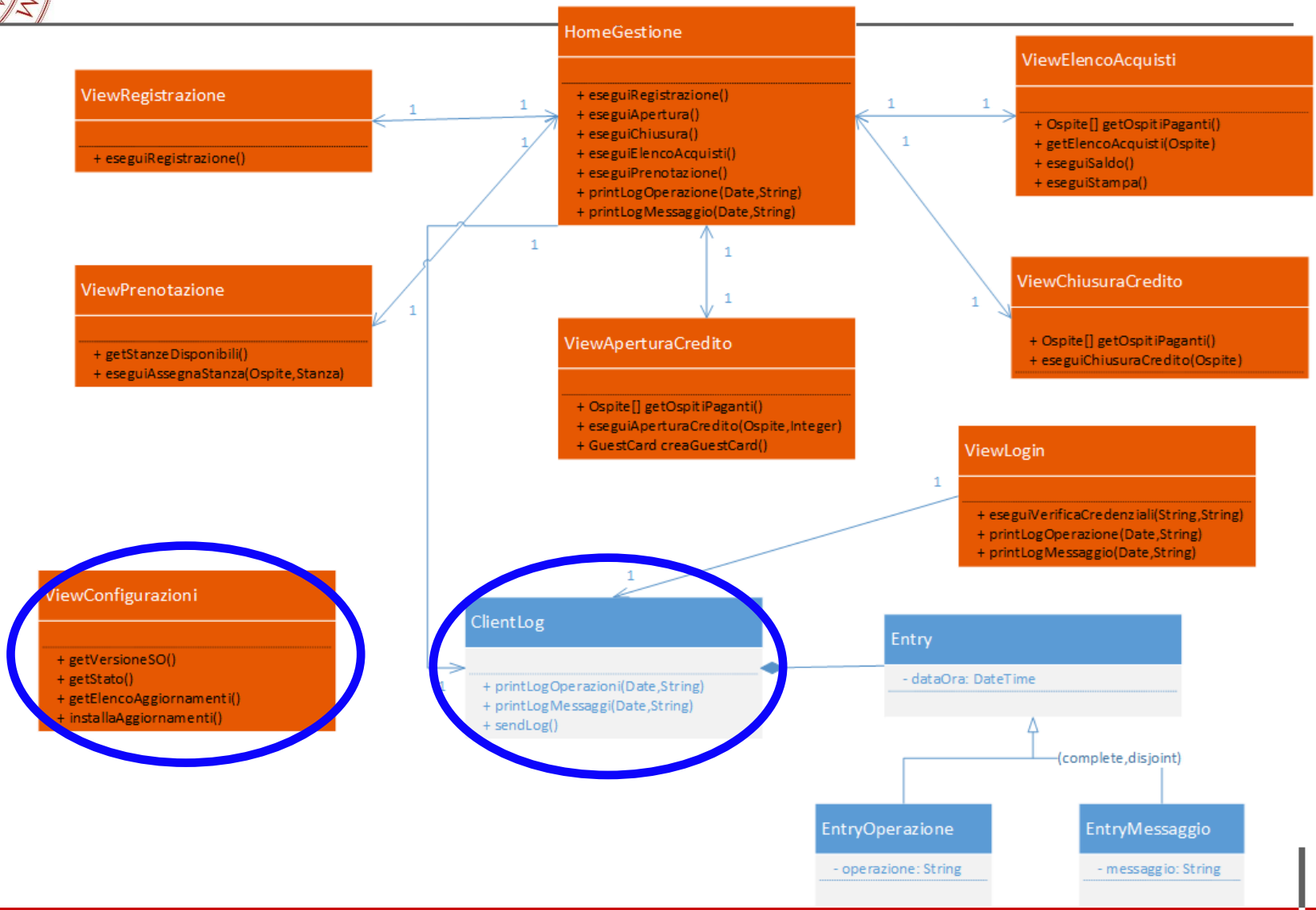
Struttura: Esempio



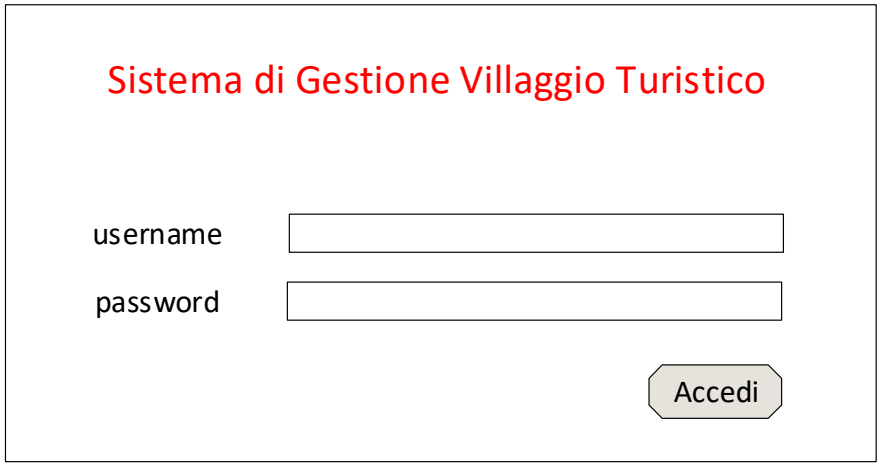
Struttura: Esempio



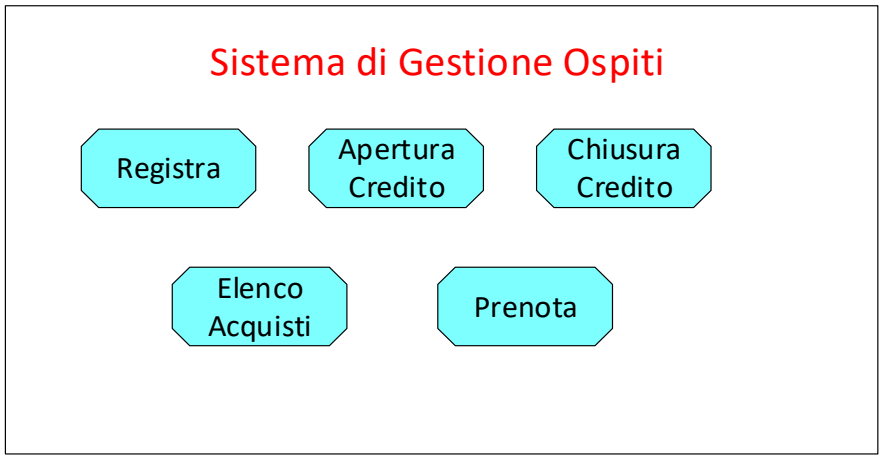
Struttura: Esempio



Struttura: Esempio



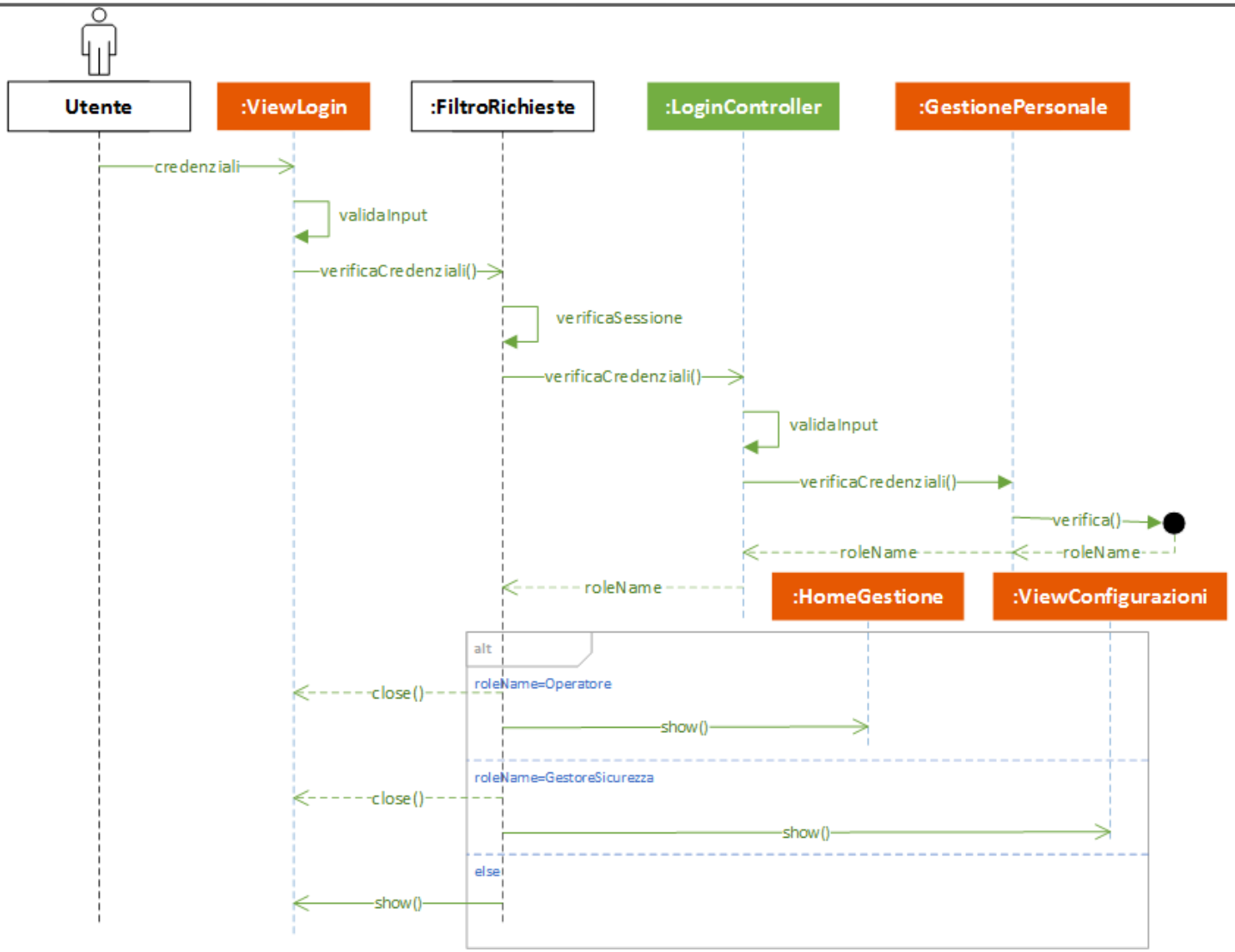
Struttura: Esempio



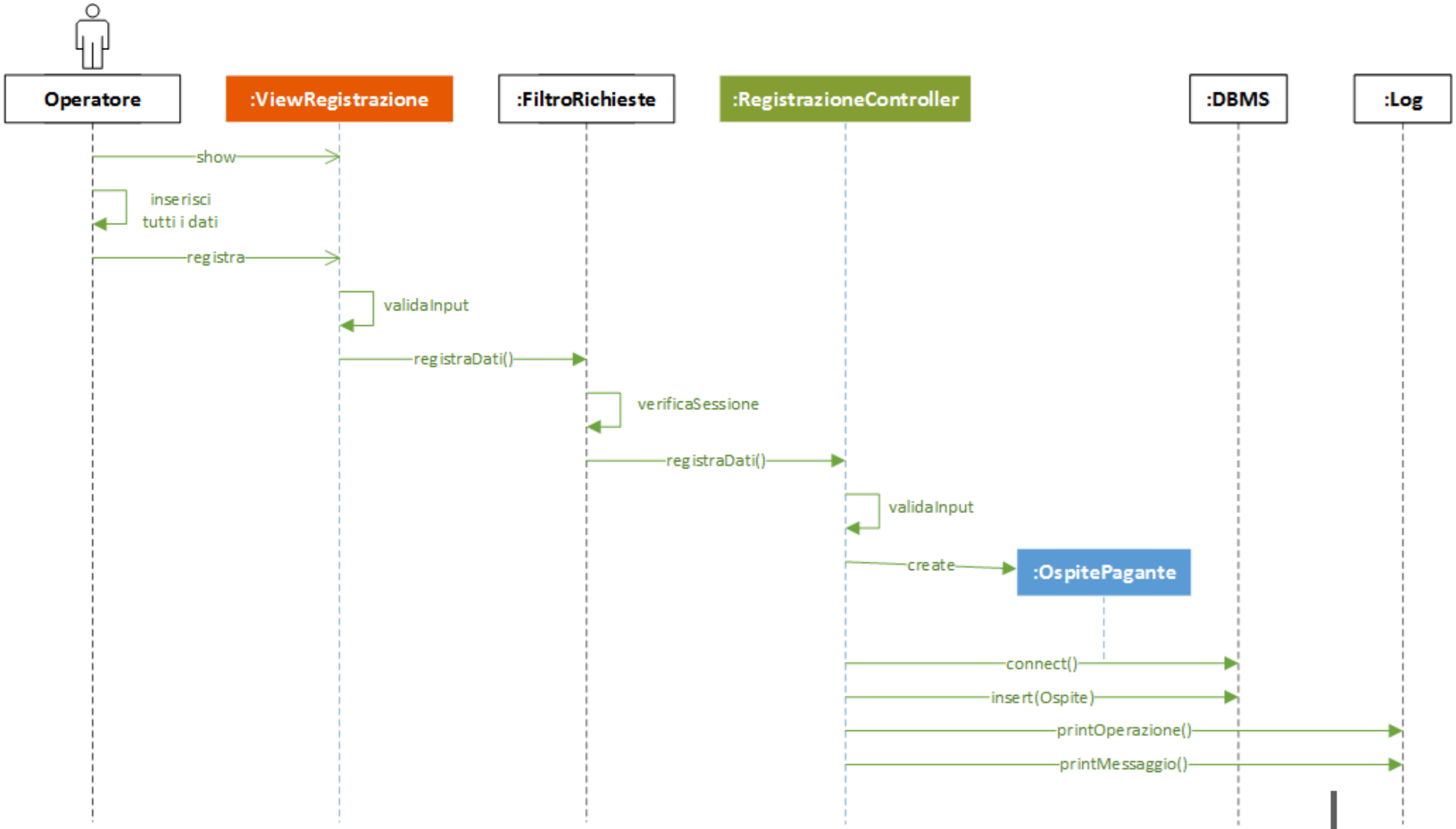
Architettura: Interazione

- Durante la Progettazione di Dettaglio della parte di Interazione è necessario
  - ridefinire i protocolli di interazione emersi in fase di Analisi dettagliandoli tenendo conto delle nuove entità emerse in progettazione
  - progettare accuratamente i protocolli di interazione verso i sistemi esterni
  - definire nuovi protocolli di interazione tra le classi che sono state introdotte nella progettazione

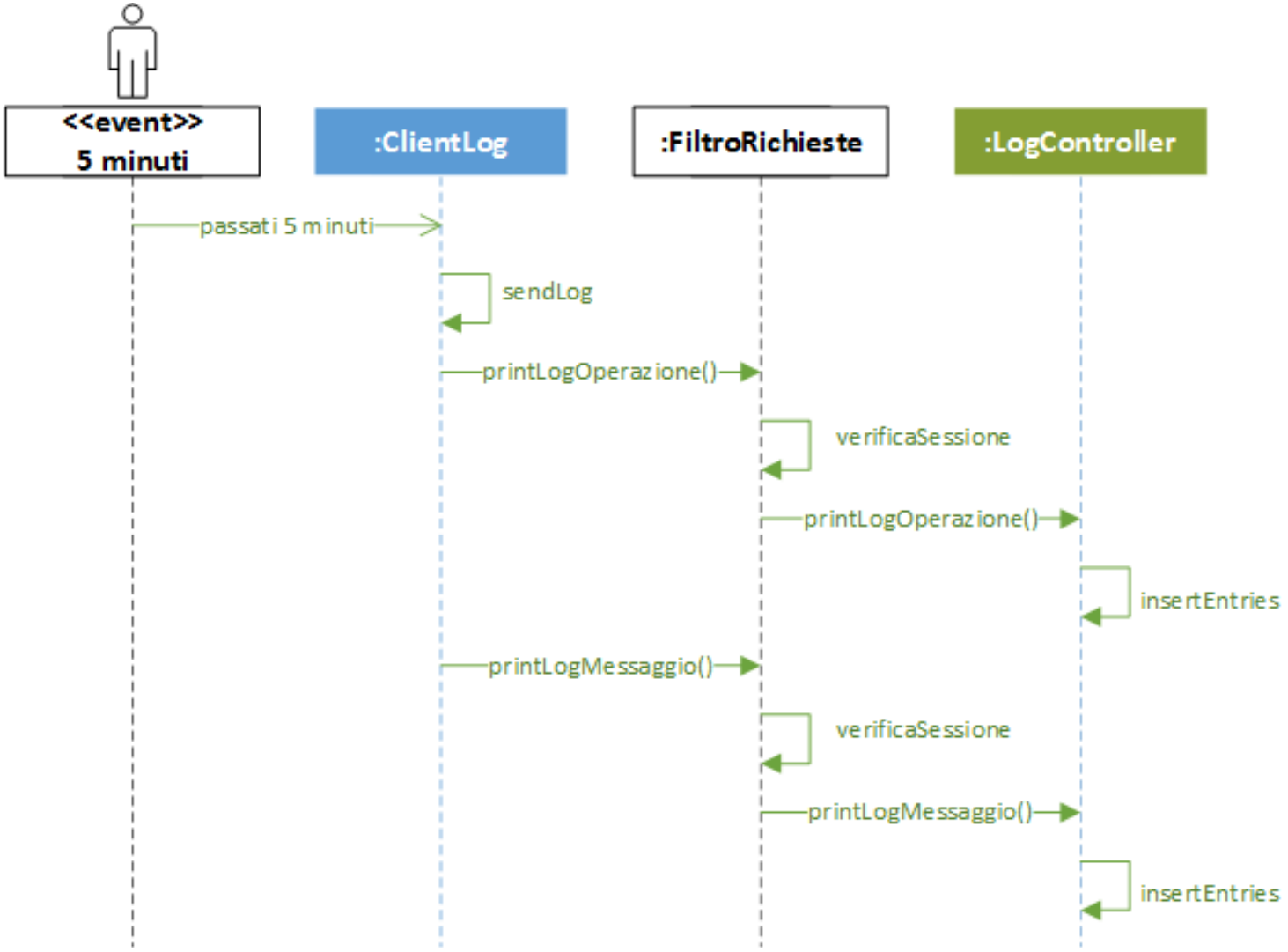
Interazione: Esempio



Interazione: Esempio



Interazione: Esempio



Architettura: Comportamento

- Durante la progettazione di dettaglio della parte di Comportamento è necessario
  - definire gli algoritmi che implementano le operazioni complesse/complicate in modo chiaro e preciso avvalendosi eventualmente di diagrammi delle attività
  - dettagliare i diagrammi di stato/attività già definiti nella fase precedente
  - eventualmente aggiungere diagrammi di stato/attività per le nuove entità emerse in questa fase

Progettazione della persistenza

Progettazione

Progettazione della Persistenza

- La persistenza dei dati è un fattore cruciale nello sviluppo di un sistema
- Il progettista dopo un'attenta valutazione di
  - vincoli imposti dai requisiti funzionali (tempi di risposta, requisiti di protezione e privacy, …)
  - tipologia di accesso accesso ai dati (lettura, scrittura, ricerche)
  - frequenza di accesso ai dati (quanto spesso devo accedere ai dati?)
  - criticità e consistenza dei dati (quanto spesso cambiano i dati? quali sono i costi di eventuale "perdite" nelle modifiche dei dati?)

  dovrà scegliere la tecnica migliore di persistenza

Progettazione della Persistenza

- Per ogni sistema va valutato attentamente quale strategia dà il miglior bilanciamento tra i vincoli e le forze in gioco nel sistema
- Non è sempre detto che l'adozione di un (R)DBMS sia la risposta corretta
- Per esempio se dobbiamo memorizzare dei log la strategia migliore è quella delle scrittura su file:
  - la maggior parte delle funzionalità "scrivono" solamente una o più righe nel log e l'accesso deve essere molto veloce → il log non deve pesare troppo nei tempi di risposta nel sistema
  - solo gli strumenti di analisi accedono in lettura al log e solitamente occorre analizzare ogni singola riga nel corretto ordine temporale, non c'è bisogno di fare ricerche

DB quando…

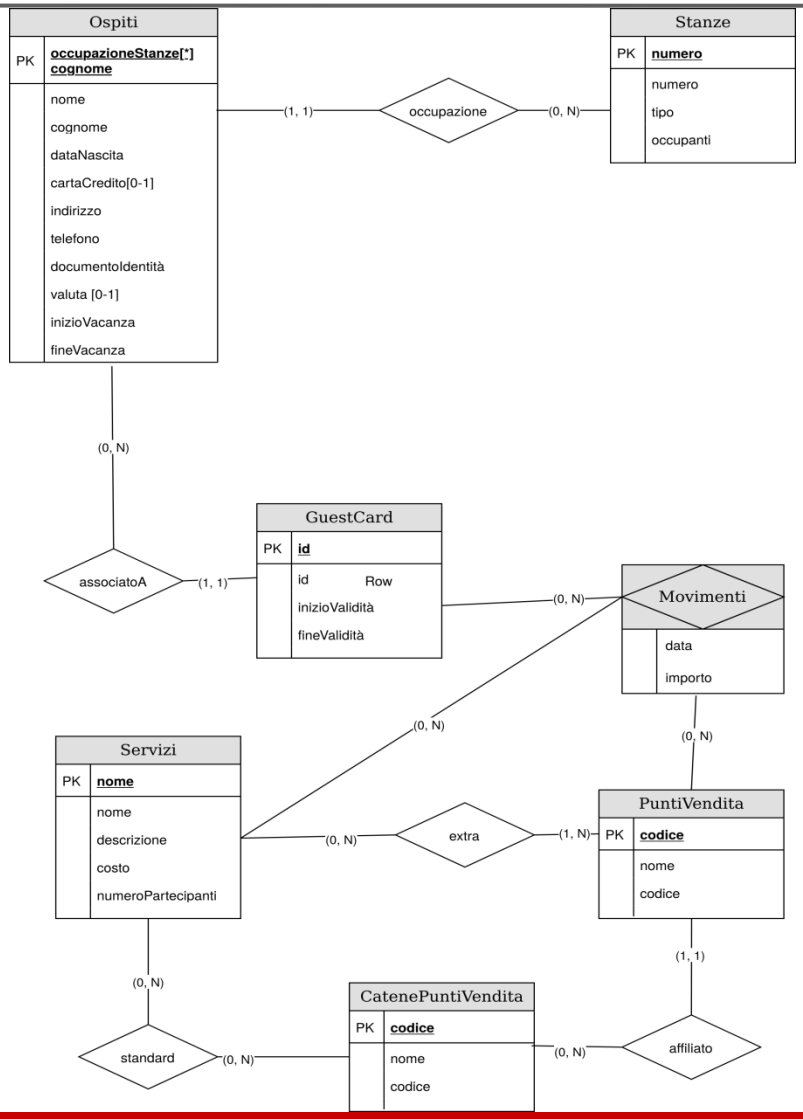
- In generale possiamo affermare che quando si ha a che fare con:
  - gestionali che trattano un numero considerevole di dati anche di natura eterogenea
  - dati che cambiano molto spesso e devono essere costantemente aggiornati
  - la “perdita” di modifiche può essere un problema
  - necessità di ripristino di versioni precedenti a seguito di un malfunzionamento

la scelta consigliata è quella di avvalersi di un DBMS

Progettazione della Persistenza

- L'output di questa fase può essere rappresentato da:
  - lo schema E-R del DB che dovrà supportare l'applicazione
  - il formato del/i file che dovranno essere scritti/letti dall'applicazione
- Sarebbe bene che sia nel caso di DB che di file ci fosse una piccola analisi del rischio per capire se
  - il DB è protetto in modo adeguato
  - il/i file necessitano di meccanismi di protezione
- Il punto di partenza di tale analisi sono i livelli di protezione e privacy richiesti per i diversi dati che saranno memorizzati

Esempio: DB Villaggio Turistico



Esempio: log Villaggio Turistico

- Formato file per Log delle operazioni

DataOra operazione esecutore

- Formato file per Log dei messaggi

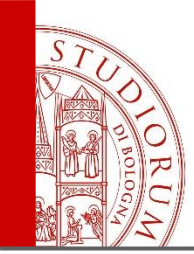
DataOra messaggio protetto invio/ricezione autore

Progettazione del collaudo

Progettazione

Progettazione del Collaudo

- La base di partenza di questa attività
  è il Piano del Collaudo sviluppato nell'analisi
- Dopo la Progettazione di Dettaglio è possibile scrivere
  i test unitari di ciascuna classe
- Successivamente vanno progettati con cura anche
  i test di integrazione del sistema
- L'output di questa attività è rappresentato
  dalla Suite completa dei test unitari e di integrazione



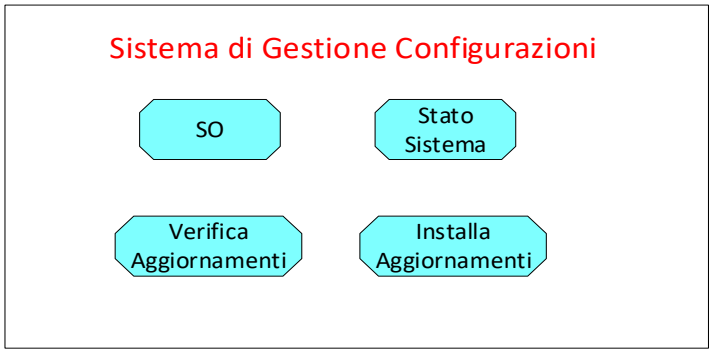
Progettazione per il deployment

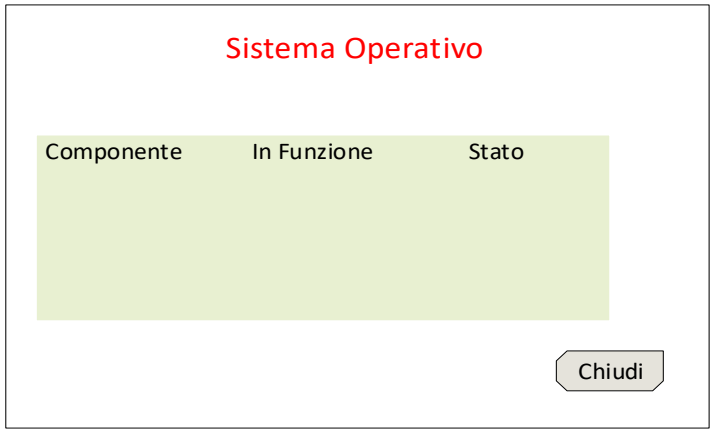
Progettazione

Progettazione per il Deployment

- Seguiremo le linee guida già viste nel blocco della sicurezza:
  1. Includere supporto per visionare e analizzare le configurazioni
  2. Minimizzare i privilegi di default
  3. Localizzare le impostazioni di configurazione
  4. Fornire modi per rimediare a vulnerabilità di sicurezza

Deployment: Esempio





Deployment: Esempio

Lato server:

- i server dovranno essere installati su macchine all'interno di una rete privata
- la rete privata dovrà essere opportunamente protetta da un firewall a cifratura di pacchetti
- l'unico punto di contatto verso l'esterno è il Broker